

Programming Assignment #4:

Returns and Conditionals

This assignment will give you greater freedom of choice. I strongly encourage you to challenge yourself – it's in writing good, difficult programs that you learn to code well.

Grading Criteria

No matter what you choose to complete, I will grade the following things:

- Helpful comments on the class and each method
- Good programming style (meaningful variable names, good indentation, etc)
- Appropriate use of returns
- Good method decomposition
- Adherence to the problem description or good pitch made prior to submission
- Construction of a single Scanner object passed to methods
- Localization of variables (tightest scope possible)
- Appropriate use of data types (int vs. double)

Generic development strategies:

- Tackle parts of the program (caloric need, caloric intake, change in weight) one at a time, rather than writing the entire program at once. Write a bit of code, get it to compile, etc.
- Write small amounts of code and then compile and test this code to make sure it works properly.
- Many students get "cannot find symbol" errors related to scope, parameters, and returns. Common mistakes are forgetting to pass / return a value, not storing a returned value into a variable, and using the wrong variable name.

Options

More details for options 2-4 are in following pages. Please read them CAREFULLY!

1. Complete returns and conditionals practice posted on the website 10/30. This is an appropriate choice if you feel you need more conceptual practice. You will need to submit a single file which each of the methods I asked you to write commented appropriately. If you complete this set, I will augment it or you can begin work on one of the following options.
2. Course grade calculator: your program will prompt the user for several grades and compute an overall grade.
3. College applicant comparison: your program will ask for test scores for two college applicants and display which one is a better candidate.
4. Calorie tracker: your program will prompt the user for what he or she has eaten and for vital statistics. Based on these, the program will tell the user how much weight they will gain or lose over time.
5. Your own design using returns, Scanner and if/else. You could use a DrawingPanel, Math... you just have to OK this with me before you get started.

Course Grade Calculator

Your program will use a student's grades on homework and exams to compute an overall course grade. The following is one example log of execution of the program (user input is underlined).

Feel free to use a particular teacher's grading scheme as long as it is reasonably complex. You will have to include what that scheme is either in the introduction or in a comment.

```
This program accepts your homework scores and
scores from two exams as input and computes
your grade in the course.
```

```
Homework and Exam 1 weights? 50 20
Using weights of 50 20 30
```

```
Homework:
```

```
Number of assignments? 3
Assignment 1 score and max? 14 15
Assignment 2 score and max? 16 20
Assignment 3 score and max? 19 25
Days participated? 4
Total points = 65 / 80
Weighted score = 40.63
```

```
Exam 1:
```

```
Score? 81
Curve? 0
Total points = 81 / 100
Weighted score = 16.2
```

```
Exam 2:
```

```
Score? 95
Curve? 10
Total points = 100 / 100
Weighted score = 30.0
```

```
Course grade = 86.83
```

The course grade is a weighted average. To compute a weighted average, the student's point scores in each category are divided by the total points for that category and multiplied by that category's weight.

Part of the homework score is determined by how many days of participation a student completes. Each day of participation is worth 4 points, up to a maximum of 20 possible section points.

In the log of execution shown, the course has 50% weight for homework, 20% weight for exam 1, and 30% weight for exam 2. There are 3 homework assignments worth 15, 20, and 25 points respectively. The student received homework scores of 14, 16, and 19, and attended 4 sections (earning 16 points for doing so). The student received an exam 1 score of 81. The student earned an exam 2 score of 95; the exam was curved by +10 points, but exam scores are capped at 100, so the student was given 100 for exam 2.

The following calculations produce the student's course grade from the above log of execution:

$$\text{Grade} = \text{WeightedHomeworkScore} + \text{WeightedExam1Score} + \text{WeightedExam2Score}$$

$$\text{Grade} = \left(\frac{14 + 16 + 19 + (4 \times 4)}{15 + 20 + 25 + 20} \times 50 \right) + \left(\frac{81}{100} \times 20 \right) + \left(\frac{100}{100} \times 30 \right)$$

$$\text{Grade} = 40.63 + 16.2 + 30.0$$

$$\text{Grade} = 86.83$$

Note that the preceding equations are not Java math. In Java, an integer expression such as 81/100 would evaluate to 0, but above the value intended is 0.81.

The program behaves differently depending on the user input it receives; you should look at all of the example logs of execution on the course web site to get a more comprehensive example of the program's behavior.

Program Behavior Details:

The program asks the user for the weights of the homework and exam 1. Using this information, the program can deduce the weight of exam 2 as 100 minus the other two weights.

You may assume that the user enters valid input. For example, assume that the user enters a number of homework assignments no less than 1, and that the sum of category weights entered will be no more than 100. The weight of a particular category (homework, exam 1, or exam 2) will be non-negative but could be 0.

You should handle the following two special cases of user input:

- A student might receive extra credit on a particular assignment, so for example 22 / 20 is a legal assignment score. But the total points for homework are capped at the maximum possible. For example if a student receives 63 total points out of a maximum of 60, your program should cap this to 60 / 60.
- The maximum score for an exam is 100. If the curved exam score exceeds 100, a score of 100 is used.

Use the `Math.max` and `Math.min` methods to constrain numbers to a given range.

Notice that all weighted scores and grades printed by the program are shown with no more than 2 digits after the decimal point.

The code to compute the student's homework scores requires you to compute a cumulative sum.

- Use static methods that accept parameters and return values where appropriate for structure and to eliminate redundancy. **For full credit, you must use at least 3 non-trivial methods other than `main` and `round2`.**
- Your `main` method should still represent a summary of the overall program; the majority of the behavior should come from other methods. To fully achieve this goal, some of your methods will need to return values back to their caller. Each method should perform a coherent task and should not do too large a share of the overall work.
- When handling numeric data, you are expected to choose appropriately between types `int` and `double`.

College Applicant Comparison

You are to write a program that prompts the user for information about two applicants and that computes an overall score for each applicant. This is a simplified version of a program that might be used for admissions purposes.

Look at the sample log of execution to see how your program is to behave. For each applicant, we prompt for exam scores (either SAT or ACT) and overall GPA. The exam information is turned into a number between 0 and 100 and the GPA information is turned into a number between 0 and 100 and these two scores are added together to get an overall score between 0 and 200. After obtaining scores for each applicant, the program reports which one looks better or whether they look equal.

Notice that the program asks for each applicant whether to enter SAT scores or ACT scores (SAT scores are integers that vary between 200 and 800, ACT scores are integers that vary between 1 and 36). In the case of SAT scores, the user is prompted for SAT math, reading, and writing scores. In the case of ACT scores, the user is prompted for English, math, reading and science scores. These scores are turned into a real-valued number between 0 and 100 using the following formulas:

For SAT Scores:

$$\frac{2 \cdot \mathit{math} + \mathit{reading} + \mathit{writing}}{32}$$

For ACT Scores:

$$\frac{\mathit{English} + 2 \cdot \mathit{math} + \mathit{reading} + \mathit{science}}{1.8}$$

After computing this exam score, we compute a number between 0 and 100 based on the GPA. The program prompts for the GPA, the maximum GPA, and a transcript multiplier. All three of these values are real values (i.e., they can have a decimal part). The transcript multiplier is a value between 0.8 and 1.0 that the admissions staff use to account for differences across students and across schools. For example, a student who takes more AP courses or a student who comes from a high school that is known to have tough grading standards will get a higher transcript multiplier. You should turn this into a score between 0 and 100 using the following formula:

$$\frac{\mathit{actual_gpa}}{\mathit{max_gpa}} \cdot 100 \cdot \mathit{transcript_multiplier}$$

At this point your program has two scores that vary from 0 to 100, one from test scores and one from GPA. The overall score for the applicant is computed as the sum of these two numbers (exam result + gpa result). Because each of these numbers is between 0 and 100, the overall score for an applicant ranges from 0 to 200.

As indicated in the sample log of execution, your program is to report the exam and GPA subscores and the overall score for each applicant. These should be rounded to two decimal places. In addition to reporting the score for each applicant, the program should also produce whichever of the following messages is appropriate:

The first applicant seems to be better
The second applicant seems to be better
The two applicants seem to be equal

You do not have to perform any error checking. We will assume that the user enters numbers and that they are in the appropriate range.

- Use static methods to eliminate redundant code and to break the problem up into logical subtasks.
- Main method should be short so that a person can easily see the overall structure of the program.
- Write at least five static methods other than main and round2 to break this problem up into smaller subtasks and you should make sure that no single method is doing too much work.
- This program involves both integer data and real data, you need to use appropriate type declarations (type int and calls on nextInt for integer data, type double and calls on nextDouble for real-valued data)

Sample log of execution (user input bold and underlined)

This program compares two applicants to determine which one seems like the stronger applicant. For each candidate I will need either SAT or ACT scores plus a weighted GPA.

Information for applicant #1:

```
do you have 1) SAT scores or 2) ACT scores? 1  
SAT math? 450  
SAT critical reading? 530  
SAT writing? 490  
exam score = 60.0  
overall GPA? 3.4  
max GPA? 4.0  
Transcript Multiplier? 0.9  
GPA score = 76.5
```

Information for applicant #2:

```
do you have 1) SAT scores or 2) ACT scores? 2  
ACT English? 25  
ACT math? 20  
ACT reading? 18  
ACT science? 15  
exam score = 54.44  
overall GPA? 3.3  
max GPA? 4.0  
Transcript Multiplier? 0.95  
GPA score = 78.38
```

First applicant overall score = 136.5
Second applicant overall score = 132.82
The first applicant seems to be better

Calorie Tracker

The program asks the user to input personal information and eating habits over the course of a day to compute how many pounds the user is gaining or losing per week. The program behaves differently depending on user input and on gender, as described in the “Daily Caloric Intake” section.

Below is an example log of execution for a female user. Look at the other example logs on the back page to get more examples of the program's behavior. User input is bold and underlined.

```
Enter information about yourself and your
diet to see if you are generally gaining or
losing weight.

Age? (years) 25
Height? (inches) 65
Weight? (pounds) 120
Level of activity? (1-5) 2
Daily caloric need = 1770.31

Breakfast:
How many things did you eat? 3
Item 1 calories? 120
Item 2 calories? 225
Item 3 calories? 100
Total calories = 445

Lunch:
How many things did you eat? 1
Item 1 calories? 750
Total calories = 750

Dinner:
How many things did you eat? 2
Item 1 calories? 320
Item 2 calories? 394
Total calories = 714

Daily caloric intake = 1909 (636.33/meal)
At this rate, you'll gain 0.28 lbs per week.
```

The program begins with an introduction message. Next it asks the user's age in years, height in inches, weight in pounds, and level of activity on a scale from 1-5, where 1 is sedentary and 5 is extremely active. This information is used to compute the user's daily caloric need.

Then the program asks the user for information regarding her three meals of the day. The program asks for the number of things she ate during a meal. For each thing eaten, the user types in the number of calories that type of food has. Use a cumulative sum to compute the calories consumed per meal.

The calories per meal are then added together to output the total calories consumed in the day (the “daily caloric intake”) and average calories per meal. Using this and the daily caloric need from earlier, the program outputs the number

of pounds the person will lose or gain per week.

The daily caloric need, calories per meal, and pounds of weight gained/lost per week are **rounded to two decimal places**.

You may assume the user enters valid input. When prompted for a value, the user will enter an integer in a proper range. The user will enter an age, height, and weight > 0, and an activity level between 1 and 5 inclusive. The number of food items per meal and the number of calories per particular will be non-negative but could be 0.

The program determines the user's fluctuation in weight by a series of calculations described in the next sections.

Daily Caloric Need

A person's *daily caloric need* is the number of calories his body needs per day to maintain its current weight. This number varies greatly from person to person, and depends on several personal factors, such as age, weight, and genetics. We cannot accurately gauge one's caloric need without fancy measurement devices, so instead we will use an estimate.

$$\text{Caloric need} = \text{Activity factor} * \text{RMR}$$

- **RMR** refers to one's resting metabolic rate. We will compute the user's RMR using the Mifflin equation:

$$\text{RMR} = 4.5 * \text{Weight} + 15.9 * \text{Height} - 5 * \text{Age} + \text{OFFSET}$$

The Mifflin equation is different for men and women by a constant offset. This offset is set to 5 to compute the RMR for a man and -161 for a woman. **Please use a class constant to set this gender offset in your program.**

- RMR only represents the calories burned while resting, so the **Activity factor** is used to adjust caloric need based on one's activity level. A sedentary person has a lower activity factor than a person who is very active. There are five activity levels as defined by Mifflin, and they are listed in the chart to the right.

It's unreasonable to expect a user to know what an "activity factor" is, so instead your program will prompt the user to rate his or her level of activity on a scale from 1 to 5. If the user enters a 1, then your program should use an activity factor of 1.2; if the user enters a 2, then your program should use a factor of 1.375; and so on.

Level of activity	Activity Factor
1	1.2
2	1.375
3	1.55
4	1.725
5	1.9

In the log of execution shown, the user is 25 years old, 5 foot 5, weighs 120 pounds, and rates her lifestyle as a 2 on the 5-point level of activity scale. The following calculations compute the daily caloric need from the log:

$$\text{RMR} = 4.5 * 120 + 15.9 * 65 - 5 * 25 + (-161) = \mathbf{1287.5}$$

$$\text{Activity factor} = \mathbf{1.375}$$
 (from activity level of 2)

Daily Caloric Intake

A person's *daily caloric intake* is the number of calories the person consumes in a day. This is simply a sum of the calories consumed for breakfast, lunch, and dinner. You should use a cumulative sum to total the calories eaten per meal.

Your program should always prompt for exactly 3 meals, and those meals should always be breakfast, lunch, and dinner. Your code does not have to be flexible about handling a different number or different types of meals.

In the log of execution shown, the user ate 3 things for breakfast, 1 thing for lunch, and 2 things for dinner. The following calculations compute the daily caloric intake from the log:

$$\text{Breakfast} = 120 + 225 + 100 = \mathbf{445}$$

$$\text{Lunch} = 750 = \mathbf{750}$$

$$\text{Dinner} = 320 + 394 = \mathbf{714}$$

Note that in Java, $1909/3$ would evaluate to 636, but the value desired is 636.33.

Change in Weight

The difference between a person's caloric intake and caloric daily needs can be used to compute the weight gained or lost over the course of a day. This difference we will call $\Delta\text{Calories}$.

$$\Delta\text{Calories} = \text{Caloric intake} - \text{Caloric need}$$

It is not very helpful to know how many calories the user gains or loses, since we measure weight in pounds. One pound equals approximately 3500 calories. It is also more interesting to output the number of pounds lost per week than per day, since the pounds per day is likely to be too small. Your program

should convert the change in calories per day to the change in pounds per week using the following formula:

$$\Delta\text{Pounds (per week)} = 7 * \Delta\text{Calories} / 3500$$

If the change is positive, then the person has eaten more calories than necessary and the body will store the unneeded calories as fat, resulting in the user gaining weight. If the change is negative, then the person has burned more calories than she consumed and she will lose weight. If the change is 0, then the person maintains exactly the same weight.

The following calculations compute the pounds gained per week from the log:

$\Delta\text{Calories} = 1909 - 1770.31 = \mathbf{138.69}$

$\Delta\text{Pounds (per day)} = 138.69 / 3500 = \mathbf{0.0396}$

Additional Output:

The following logs show the program in two other scenarios. Log #2 has the same input as #1, but the formula offset is for males. Log #3 shows that if the user's $\Delta\text{Pounds per week}$ is 0, the user maintains her weight. These logs do not cover all possible cases; **see the course web site for additional logs** and make sure you match all of their output before turning in.

Log #2: Male (OFFSET = 5)	Log #3: Female (OFFSET = -161)
Enter information about yourself and your diet to see if you are generally gaining or losing weight. Age? (years) <u>25</u> Height? (inches) <u>65</u> Weight? (pounds) <u>120</u> Level of activity? (1-5) <u>2</u> Daily caloric need = 1998.56 Breakfast: How many things did you eat? <u>3</u> Item 1 calories? <u>120</u> Item 2 calories? <u>225</u> Item 3 calories? <u>100</u> Total calories = <u>445</u> Lunch: How many things did you eat? <u>1</u> Item 1 calories? <u>750</u> Total calories = <u>750</u> Dinner: How many things did you eat? <u>2</u> Item 1 calories? <u>320</u> Item 2 calories? <u>394</u> Total calories = <u>714</u> Daily caloric intake = 1909 (636.33/meal) At this rate, you'll lose 0.18 lbs per week.	Enter information about yourself and your diet to see if you are generally gaining or losing weight. Age? (years) <u>81</u> Height? (inches) <u>44</u> Weight? (pounds) <u>159</u> Level of activity? (1-5) <u>1</u> Daily caloric need = 1018.92 Breakfast: How many things did you eat? <u>0</u> Total calories = 0 Lunch: How many things did you eat? <u>2</u> Item 1 calories? <u>425</u> Item 2 calories? <u>575</u> Total calories = 1000 Dinner: How many things did you eat? <u>1</u> Item 1 calories? <u>18</u> Total calories = 18 Daily caloric intake = 1018 (339.33/meal) At this rate, you'll maintain your weight.